Enhanced Testing Implementation

===== TESTING COMMON PATTERNS =====

Testing NaiveMatcher

Text length: 62770 characters

Pattern: "the" (length: 3)

Found 1055 matches

First 5 matches at positions:

295 445 543 551 788

Time taken: 36954539 nanoseconds

Character comparisons: 68964

Testing NaiveMatcher

Text length: 62770 characters

Pattern: "and" (length: 3)

Found 470 matches

First 5 matches at positions:

518 931 962 1471 1557

Time taken: 35013906 nanoseconds

Character comparisons: 67430

Testing NaiveMatcher

Text length: 62770 characters

Pattern: "in" (length: 2)

Found 867 matches

First 5 matches at positions:

23 67 78 90 93

Time taken: 3833110 nanoseconds

Character comparisons: 66236

Testing NaiveMatcher

Text length: 62770 characters

Pattern: "algorithm" (length: 9)

Found 0 matches

First 5 matches at positions:

Time taken: 6328071 nanoseconds

Character comparisons: 66952

===== TESTING WITH SMALL TEXT =====

Testing NaiveMatcher

Text length: 666 characters

Pattern: "the" (length: 3)

Found 3 matches

First 5 matches at positions:

199 322 629

Time taken: 47328 nanoseconds

Character comparisons: 735

Testing NaiveMatcher

Text length: 666 characters

Pattern: "and" (length: 3)

Found 1 matches

First 5 matches at positions:

638

Time taken: 86555 nanoseconds

Character comparisons: 715

Testing NaiveMatcher

Text length: 666 characters

Pattern: "in" (length: 2)

Found 12 matches

First 5 matches at positions:

33 42 86 128 196

Time taken: 63728 nanoseconds

Character comparisons: 713

Testing NaiveMatcher

Text length: 666 characters

Pattern: "algorithm" (length: 9)

Found 7 matches

First 5 matches at positions:

46 110 249 259 269

Time taken: 62074 nanoseconds

Character comparisons: 758

===== VERIFICATION TEST =====

Sample text: abababababa

Sample pattern: aba

Testing NaiveMatcher

Text length: 9 characters

Pattern: "aba" (length: 3)

Found 4 matches

First 5 matches at positions:

0 2 4 6

Time taken: 10697 nanoseconds

Character comparisons: 15

===== PATTERN TYPE TESTS =====

==== PATTERN TYPE TESTS ====

Testing NaiveMatcher

Text length: 62770 characters

Pattern: "the" (length: 3)

Found 1055 matches

First 5 matches at positions:

295 445 543 551 788

Time taken: 34003691 nanoseconds

Character comparisons: 68964

Testing NaiveMatcher

Text length: 62770 characters

Pattern: "xyzabc123" (length: 9)

Found 0 matches

First 5 matches at positions:

Time taken: 11482163 nanoseconds

Character comparisons: 62823

Testing NaiveMatcher

Text length: 62770 characters

Pattern: "implementation of string matching algorithms" (length: 44)

Found 0 matches

First 5 matches at positions:

Time taken: 46247976 nanoseconds

Character comparisons: 66367

Testing NaiveMatcher

Text length: 62770 characters

Pattern: "abababababab" (length: 10)

Found 0 matches

First 5 matches at positions:

Time taken: 5548765 nanoseconds

Character comparisons: 66666

===== COMPARING ALGORITHMS =====

==== ALGORITHM COMPARISON ====

Text length: 62770

Pattern: "the" (length: 3)

NaiveMatcher:

Matches found: 1055

Time: 4123074 ns

Comparisons: 68964

KMPMatcher:

Matches found: 1055

Time: 15677436 ns

Comparisons: 62772

BoyerMooreMatcher:

Matches found: 1055

Time: 29284317 ns

Comparisons: 26330

RabinKarpMatcher:

Matches found: 1055

Time: 69896410 ns

Comparisons: 3535

==== ALGORITHM COMPARISON ====

Text length: 62770

Pattern: "algorithm" (length: 9)

NaiveMatcher:

Matches found: 0

Time: 15243656 ns

Comparisons: 66952

KMPMatcher:

Matches found: 0

Time: 7900514 ns

Comparisons: 62778

BoyerMooreMatcher:

Matches found: 0

Time: 3390660 ns

Comparisons: 8941

RabinKarpMatcher:

Matches found: 0

Time: 6147603 ns

Comparisons: 706

==== ALGORITHM COMPARISON ====

Text length: 62770

Pattern: "abababababab" (length: 10)

NaiveMatcher:

Matches found: 0

Time: 48806502 ns

Comparisons: 66666

KMPMatcher:

Matches found: 0

Time: 8143825 ns

Comparisons: 62781

BoyerMooreMatcher:

Matches found: 0

Time: 1877966 ns

Comparisons: 6736

RabinKarpMatcher:

Matches found: 0

Time: 6180890 ns

Comparisons: 663

===== TESTING BOYER-MOORE STRENGTHS =====

Testing with long pattern:

==== ALGORITHM COMPARISON ====

Text length: 62770

Pattern: "implementation of string matching algorithms" (length: 44)

NaiveMatcher:

Matches found: 0

Time: 2162784 ns

Comparisons: 66367

KMPMatcher:

Matches found: 0

Time: 10720806 ns

Comparisons: 62817

BoyerMooreMatcher:

Matches found: 0

Time: 1052553 ns

Comparisons: 3958

RabinKarpMatcher:

Matches found: 0

Time: 5105016 ns

Comparisons: 628

Testing with rare-ended pattern:

==== ALGORITHM COMPARISON ====

Text length: 62770

Pattern: "theXYZ" (length: 6)

NaiveMatcher:

Matches found: 0

Time: 862664 ns

Comparisons: 70016

KMPMatcher:
Matches found: 0
Time: 22184797 ns
Comparisons: 62775

BoyerMooreMatcher:
Matches found: 0
Time: 3087915 ns
Comparisons: 11354

RabinKarpMatcher:
Matches found: 0
Time: 5017585 ns
Comparisons: 833
Large text file not available. Skipping test.

===== TESTING RABIN-KARP STRENGTHS =====

Testing with multiple patterns:

==== ALGORITHM COMPARISON ====
Text length: 62770
Pattern: "the" (length: 3)

NaiveMatcher:
Matches found: 1055
Time: 2753258 ns
Comparisons: 68964

KMPMatcher:

Matches found: 1055

Time: 12875340 ns

Comparisons: 62772

BoyerMooreMatcher:

Matches found: 1055

Time: 7631408 ns

Comparisons: 26330

RabinKarpMatcher:

Matches found: 1055

Time: 7227513 ns

Comparisons: 3535

==== ALGORITHM COMPARISON ====

Text length: 62770

Pattern: "and" (length: 3)

NaiveMatcher:

Matches found: 470

Time: 1259874 ns

Comparisons: 67430

KMPMatcher:

Matches found: 470

Time: 25428969 ns

Comparisons: 62772

BoyerMooreMatcher:

Matches found: 470

Time: 4040069 ns

Comparisons: 23821

RabinKarpMatcher:

Matches found: 470

Time: 1618355 ns

Comparisons: 2619

==== ALGORITHM COMPARISON ====

Text length: 62770

Pattern: "for" (length: 3)

NaiveMatcher:

Matches found: 132

Time: 1347211 ns

Comparisons: 64096

KMPMatcher:

Matches found: 132

Time: 16782830 ns

Comparisons: 62772

BoyerMooreMatcher:

Matches found: 132

Time: 3258094 ns

Comparisons: 23846

RabinKarpMatcher:

Matches found: 132

Time: 2251042 ns

Comparisons: 992

==== ALGORITHM COMPARISON ====

Text length: 62770

Pattern: "algorithm" (length: 9)

NaiveMatcher:

Matches found: 0

Time: 3575557 ns

Comparisons: 66952

KMPMatcher:

Matches found: 0

Time: 13371517 ns

Comparisons: 62778

BoyerMooreMatcher:

Matches found: 0

Time: 2650446 ns

Comparisons: 8941

RabinKarpMatcher:

Matches found: 0

Time: 1161444 ns

Comparisons: 706

Testing patterns prone to hash collision:

==== ALGORITHM COMPARISON ====

Text length: 62770

Pattern: "abcdef" (length: 6)

NaiveMatcher:

Matches found: 0

Time: 1128675 ns

Comparisons: 66668

KMPMatcher:

Matches found: 0

Time: 14780594 ns

Comparisons: 62775

BoyerMooreMatcher:

Matches found: 0

Time: 1108965 ns

Comparisons: 12210

RabinKarpMatcher:

Matches found: 0

Time: 1207532 ns

Comparisons: 596

==== ALGORITHM COMPARISON ====

Text length: 62770

Pattern: "fedcba" (length: 6)

NaiveMatcher:

Matches found: 0

Time: 1829583 ns

Comparisons: 64034

KMPMatcher:

Matches found: 0

Time: 19188168 ns

Comparisons: 62775

BoyerMooreMatcher:

Matches found: 0

Time: 1459924 ns

Comparisons: 12126

RabinKarpMatcher:

Matches found: 0

Time: 5176425 ns

Comparisons: 572

==========

SCALABILITY TESTS - 150, 10K, 100K, 1M CHARACTERS

==========

==========

TESTING WITH 150 CHARACTERS

==========

Text size: 150 characters

--- Pattern: "the" (3 chars) ---

Algorithm	Matches	Comparisons	Time(ms)	Comp/ms	Efficiency
NaiveMatcher	0	152	0 ms	0	★★ FAST
KMPMatcher	0	152	0 ms	0	★★ FAST
BoyerMooreMatcher	0	51	0 ms	0	★★ FAST
RabinKarpMatcher	0	0	0 ms	0	★★★ BEST

Pattern analysis:
- Length ratio: 0.0200 (pattern/text)
- Repetitive: No
- Long pattern: No
- Predicted best: Naive (small input)

--- Pattern: "algorithm" (9 chars) ---

Algorithm	Matches	Comparisons	Time(ms)	Comp/ms	Efficiency
NaiveMatcher	0	149	0 ms	0	★★ FAST
KMPMatcher	0	158	0 ms	0	★★ FAST
BoyerMooreMatcher	0	19	0 ms	0	★★ FAST
RabinKarpMatcher	0	0	0 ms	0	★★★ BEST

Pattern analysis:
- Length ratio: 0.0600 (pattern/text)
- Repetitive: No
- Long pattern: No
- Predicted best: Naive (small input)

--- Pattern: "string matching" (15 chars) ---

Algorithm	Matches	Comparisons	Time(ms)	Comp/ms	Efficiency
NaiveMatcher	0	141	0 ms	0	★★ FAST

KMPMatcher 0 164 0 ms 0 ★★ FAST

BoyerMooreMatcher 0 17 0 ms 0 ★★ FAST

RabinKarpMatcher 0 1 0 ms 0 ★★★ BEST

Pattern analysis:

- Length ratio: 0.1000 (pattern/text)
- Repetitive: No
- Long pattern: No
- Predicted best: Rabin-Karp (efficient comparisons)

==========

TESTING WITH 10,000 CHARACTERS

==========

Text size: 10,000 characters

--- Pattern: "the" (3 chars) ---

Algorithm Matches Comparisons Time(ms) Comp/ms Efficiency

NaiveMatcher 146 10,912 0 ms 0 ★★ FAST

KMPMatcher 146 10,002 3 ms 3334 ○ OK

BoyerMooreMatcher 146 4,113 0 ms 0 ★★ FAST

RabinKarpMatcher 146 483 1 ms 483 ★★ EFFICIENT

Pattern analysis:

- Length ratio: 0.0003 (pattern/text)
- Repetitive: No
- Long pattern: No
- Predicted best: Rabin-Karp (efficient comparisons)

--- Pattern: "algorithm implementation" (24 chars) ---

Algorithm Matches Comparisons Time(ms) Comp/ms Efficiency

NaiveMatcher 0 10,612 0 ms 0 ★★ FAST

KMPMatcher 0 10,024 1 ms 10024 ○ OK

BoyerMooreMatcher 0 861 0 ms 0 ★★ FAST

RabinKarpMatcher 0 88 0 ms 0 ★★★ BEST

Pattern analysis:

- Length ratio: 0.0024 (pattern/text)
- Repetitive: No
- Long pattern: No
- Predicted best: Boyer-Moore (medium-long pattern)

--- Pattern: "comprehensive analysis of s..." (52 chars) ---

Algorithm Matches Comparisons Time(ms) Comp/ms Efficiency

NaiveMatcher 0 10,195 0 ms 0 ★★ FAST

KMPMatcher 0 10,052 1 ms 10052 ○ OK

BoyerMooreMatcher 0 603 0 ms 0 ★★ FAST

RabinKarpMatcher 0 111 0 ms 0 ★★★ BEST

Pattern analysis:

- Length ratio: 0.0052 (pattern/text)
- Repetitive: No
- Long pattern: Yes
- Predicted best: Boyer-Moore (medium-long pattern)

--- Pattern: "abcabcabcabcabcabcabcabcabc..." (90 chars) ---

Algorithm Matches Comparisons Time(ms) Comp/ms Efficiency

NaiveMatcher 0 10,504 0 ms 0 ★★ FAST

KMPMatcher 0 10,089 3 ms 3363 ○ OK

BoyerMooreMatcher 0 124 0 ms 0 ★★ FAST

RabinKarpMatcher 0 95 3 ms 32 ★★ EFFICIENT

Pattern analysis:

- Length ratio: 0.0090 (pattern/text)
- Repetitive: Yes
- Long pattern: Yes
- Predicted best: KMP (repetitive pattern)

==========

TESTING WITH 100,000 CHARACTERS

==========

Text size: 100,000 characters

--- Pattern: "the" (3 chars) ---

Algorithm Matches Comparisons Time(ms) Comp/ms Efficiency

NaiveMatcher 1698 109,791 2 ms 54896 ★ GOOD

KMPMatcher 1698 100,002 1 ms 100002 ★★ FAST

BoyerMooreMatcher 1698 41,912 4 ms 10478 ○ OK

RabinKarpMatcher 1698 5,682 22 ms 258 ★★ EFFICIENT

Pattern analysis:

- Length ratio: 0.0000 (pattern/text)
- Repetitive: No
- Long pattern: No
- Predicted best: Rabin-Karp (efficient comparisons)

--- Pattern: "algorithm" (9 chars) ---

Algorithm Matches Comparisons Time(ms) Comp/ms Efficiency

NaiveMatcher 0 106,728 3 ms 35576 ○ OK

KMPMatcher 0 100,008 3 ms 33336 ○ OK

BoyerMooreMatcher 0 14,253 1 ms 14253 ★★ FAST

RabinKarpMatcher 0 1,145 2 ms 573 ★★ EFFICIENT

Pattern analysis:

- Length ratio: 0.0001 (pattern/text)
- Repetitive: No
- Long pattern: No
- Predicted best: Rabin-Karp (efficient comparisons)

--- Pattern: "string matching algorithms ..." (47 chars) ---

Algorithm Matches Comparisons Time(ms) Comp/ms Efficiency

NaiveMatcher 0 106,047 1 ms 106047 ○ OK

KMPMatcher 0 100,048 1 ms 100048 ○ OK

BoyerMooreMatcher 0 6,508 0 ms 0 ★★ FAST

RabinKarpMatcher 0 1,084 1 ms 1084 ★★ EFFICIENT

Pattern analysis:

- Length ratio: 0.0005 (pattern/text)
- Repetitive: No
- Long pattern: No
- Predicted best: Boyer-Moore (medium-long pattern)

--- Pattern: "abababababababababababababa..." (100 chars) ---

Algorithm Matches Comparisons Time(ms) Comp/ms Efficiency

NaiveMatcher 0 106,150 2 ms 53075 ○ OK

KMPMatcher 0 100,101 14 ms 7150 ○ OK

BoyerMooreMatcher 0 1,076 0 ms 0 ★★★ BEST

RabinKarpMatcher 0 1,002 1 ms 1002 ★★ EFFICIENT

Pattern analysis:

- Length ratio: 0.0010 (pattern/text)
- Repetitive: Yes
- Long pattern: Yes
- Predicted best: KMP (repetitive pattern)

--- Pattern: "resolve to render unto all ..." (150 chars) ---

Algorithm	Matches	Comparisons	Time(ms)	Comp/ms	Efficiency
NaiveMatcher	1	105,839	2 ms	52920	○ OK
KMPMatcher	1	100,157	20 ms	5008	○ OK
BoyerMooreMatcher	1	4,774	0 ms	0	★★ FAST
RabinKarpMatcher	1	1,220	4 ms	305	★★ EFFICIENT

Pattern analysis:

- Length ratio: 0.0015 (pattern/text)
- Repetitive: No
- Long pattern: Yes
- Predicted best: Boyer-Moore (long pattern)

==========

TESTING WITH 1,000,000 CHARACTERS

==========

Text size: 1,000,000 characters

--- Pattern: "the" (3 chars) ---

Algorithm	Matches	Comparisons	Time(ms)	Comp/ms	Efficiency
NaiveMatcher	16807	1,098,687	51 ms	21543	○ OK

KMPMatcher 16807 1,000,002 265 ms 3774 ○ OK

BoyerMooreMatcher 16807 419,507 57 ms 7360 ○ OK

RabinKarpMatcher 16807 56,314 13 ms 4332 ★★★ BEST

Pattern analysis:

- Length ratio: 0.0000 (pattern/text)
- Repetitive: No
- Long pattern: No
- Predicted best: Rabin-Karp (efficient comparisons)

--- Pattern: "algorithm implementation" (24 chars) ---

Algorithm Matches Comparisons Time(ms) Comp/ms Efficiency

NaiveMatcher 0 1,066,742 10 ms 106674 ★ GOOD

KMPMatcher 0 1,000,024 9 ms 111114 ★★ FAST

BoyerMooreMatcher 0 85,730 9 ms 9526 ★★ FAST

RabinKarpMatcher 0 10,172 18 ms 565 ★★ EFFICIENT

Pattern analysis:

- Length ratio: 0.0000 (pattern/text)
- Repetitive: No
- Long pattern: No
- Predicted best: Boyer-Moore (medium-long pattern)

--- Pattern: "comprehensive performance a..." (73 chars) ---

Algorithm Matches Comparisons Time(ms) Comp/ms Efficiency

NaiveMatcher 0 1,023,499 11 ms 93045 ○ OK

KMPMatcher 0 1,000,075 9 ms 111119 ★ GOOD

BoyerMooreMatcher 0 57,200 5 ms 11440 ★★ FAST

RabinKarpMatcher 0 9,754 13 ms 750 ★★ EFFICIENT

Pattern analysis:
  - Length ratio: 0.0001 (pattern/text)
  - Repetitive: No
  - Long pattern: Yes
  - Predicted best: Boyer-Moore (medium-long pattern)

--- Pattern: "xyzxyzxyzxyzxyzxyzxyzxyzxyz..." (150 chars) ---

Algorithm	Matches	Comparisons	Time(ms)	Comp/ms	Efficiency
NaiveMatcher	0	1,000,822	11 ms	90984	○ OK
KMPMatcher	0	1,000,149	8 ms	125019	★ GOOD
BoyerMooreMatcher	0	6,778	5 ms	1356	★★★ BEST
RabinKarpMatcher	0	10,635	10 ms	1064	★ GOOD

Pattern analysis:
  - Length ratio: 0.0002 (pattern/text)
  - Repetitive: Yes
  - Long pattern: Yes
  - Predicted best: Boyer-Moore (long pattern)

--- Pattern: "01010101010101010101010101010..." (200 chars) ---

Algorithm	Matches	Comparisons	Time(ms)	Comp/ms	Efficiency
NaiveMatcher	0	999,897	17 ms	58817	○ OK
KMPMatcher	0	1,000,199	14 ms	71443	○ OK
BoyerMooreMatcher	0	5,002	0 ms	0	★★★ BEST
RabinKarpMatcher	0	10,290	14 ms	735	○ OK

Pattern analysis:
  - Length ratio: 0.0002 (pattern/text)
  - Repetitive: Yes

- Long pattern: Yes
- Predicted best: Boyer-Moore (long pattern)

--- Pattern: "an whose soul is absorbed i..." (200 chars) ---

Algorithm	Matches	Comparisons	Time(ms)	Comp/ms	Efficiency
NaiveMatcher	15	1,080,004	57 ms	18947	○ OK
KMPMatcher	15	1,000,209	14 ms	71444	○ OK
BoyerMooreMatcher	15	40,115	5 ms	8023	★★ FAST
RabinKarpMatcher	15	13,559	11 ms	1233	★★ EFFICIENT

Pattern analysis:
- Length ratio: 0.0002 (pattern/text)
- Repetitive: No
- Long pattern: Yes
- Predicted best: Boyer-Moore (long pattern)

==========

SCALABILITY ANALYSIS SUMMARY

==========

Key Findings:

• Small texts (150 chars): Naive often fastest due to low overhead

• Medium texts (10K chars): Algorithm differences become apparent

• Large texts (100K chars): Boyer-Moore and Rabin-Karp advantages clear

• Very large texts (1M chars): Theoretical complexities fully manifest

Recommendations by text size:

• < 1K characters: Naive for simplicity

• 1K - 50K characters: Boyer-Moore for long patterns, KMP for repetitive

• > 50K characters: Boyer-Moore or Rabin-Karp depending on pattern type

★★★ = Best overall, ★★ = Excellent in category, ★ = Good, ○ = Acceptable

Process finished with exit code 0